Reliability in Volunteer Computing Micro-blogging Services

Christopher Bayliss[1], Javier Panadero[1], Laura Calvet[1,2], and Joan Manuel Marquès[1]
[1]IN3 – Computer Science Dept., Universitat Oberta de Catalunya, Spain
[2]Valencian International University, Spain

November 4, 2020

Abstract

Volunteer computing is a type of large-scale heterogeneous distributed system where resources necessary to run the system are donated by volunteers. A draw back of volunteer computing is the unreliability of the donated resources, so redundancy is required to guarantee the fulfillment of tasks or the availability of data. In this work we consider the problem of designing a directory service policy for a Distributed Volunteer Computing Micro-Blogging Service (DVCMBS). In such services, nodes donate storage space (repositories), which is managed by a centralized directory service that decides which nodes will store replicas of blogs, ensuring their online availability when bloggers are offline. Since nodes are under no obligation to remain online, the task of a DVCMBS directory service is to allocate blog replicas to online repositories, such that the rate of availability of all blogs is maximized. At the same time, since donated storage resources are limited and directory service operations use processing resources, minimizing the number of blog replicas generated (*i.e.*, the inefficiency of the directory service) is critical. We present a simulation model of a DVCMBS, which uses a probabilistic sort and select approach to select host repositories for replicas and blogs to replicate. Exhaustive computational experiments analyze the trade-off between blog replica availability and efficiency, and identify the efficient directory service policies, with respect to availability and efficiency maximization.

Keywords: Micro-blogging services, volunteer computing, complex system reliability, discrete-event simulation.

1 Introduction

Online social media has been gaining popularity during the last decades, especially micro-blogging online social networks. These micro-blogging systems allow bloggers to post short messages, share interests, and communicate with each other. Currently, there are nearly 1 billion monthly active bloggers between Twitter, Tumblr, and other social networks [1, 2]. Despite their rapid expansion, most popular micro-blogging systems have a poor reputation in terms of protecting the privacy of the blogger's data [3]. Indeed, many experts propose designing decentralized social networks, mainly for privacy reasons [4].

Volunteer Computing (VC) [5] systems are large-scale heterogeneous distributed systems where resources (nodes) are donated by volunteers. Public contributors share a part of their idle computational resources to execute computationally expensive applications. This kind of computation has become increasingly popular due to the fact that it provides a scalable, elastic, practical, and low cost platform to meet the computational and storage demands of many applications. However, the nodes are provided in a voluntary way, which means that they may suffer from a lack of reliability, since they are usually non-dedicated and dynamic. In particular, the main characteristics of this paradigm are: a) no central authority is responsible for providing the required computational resources; b) heterogeneous (software and hardware) and low capacity computer resources spread across the Internet in contrast with few high capacity cluster of computers of commercial social networks; and c) the computational infrastructure belongs to the blogger's/node providers and is shared to build the computational infrastructure. Regarding reliability and QoS, these computing systems have to guarantee to bloggers: a) availability, *i.e.*, they can access data anytime from anywhere; b) freshness, *i.e.*, they get up-to-date data; and c) immediateness, *i.e.*, they obtain the data in a time that is felt as immediate. The fact that nodes have an uncertain behaviour regarding connection, disconnection, capacity and failure is the main challenge.

A hybrid peer-to-peer (P2P) [6] is a model that combines P2P and client-server models. Usually it has a centralized service that helps peers to locate other peers. This centralized components helps hybrid models provide a good performance at the same time that the overall system benefits from the decentralized communication between peers.

In this context, we address the problem of developing blog replica decision policies for the directory service of a distributed VC micro-blogging platform, where the directory service is the centralized component of a hybrid p2p infrastructure formed by peers voluntarily contributed by participants in the social network. The main goal is to maximize the system's reliability, in terms of online content availability, by relying on optimization techniques and statistics. Reliability is defined as 1-probability of failure. In the current context, the probability of failure is defined as the rate at which content is unavailable when it is required. The secondary goal is to make efficient use of the resources, by trying to minimize the total number of blog replicas generated. Notice that although the main use of VC is associated with high-throughput computing, we are using the VC as a storage platform, this work proposes effective decision policies which guarantee data availability.

The main contributions of this work are: 1) a mathematical model of an example VC micro-blogging service; 2) a proposed event-based simulation model of the VC micro-blogging platform, modeling both connection and disconnection of nodes along with blogger messaging behavior; 3) a proposed sort and select directory service replication decision mechanism based on probabilistic sorting criteria; 4) an exhaustive investigation of different configuration and parameter choices for the directory service replication policy; and 5) a description of the insights yielded regarding reliable and efficient VC micro-blogging services.

Our proposed directory service uses a sort and select approach for making replication decisions. There are two different types replication decisions: 1) selecting repositories in which to store replicas; and 2) selecting replicas to generate and store in a given repository. We propose probabilistic criteria for sorting candidate replication decisions. These criteria are based on estimating the probabilities that donated repositories hosting blog replicas remain available online before the associated bloggers return online after leaving once. This approach is based on the assumption that each node stores a copy of the blogs of bloggers using that node, so that if their content is lost when the node is offline, additional replicas can be generated when it arrives back online. This approach aims at maximizing the availability (of critical) content while the associated bloggers are offline. The effectiveness of the proposed sorting criteria is demonstrated by comparing directory service performance levels for replication decision selections based on greedy, biased randomized [7] and random approaches. In computational experiments, we analyze the trade-off between blog replica availability and the total number of replicas generated and identify the non-dominated directory service policies.

The remainder of this paper is structured as follows. Section 2 reviews related research and discusses the original contributions of this work. Section 3 provides a general description of the problem studied, while Section 4 presents a mathematical model for a particular example of DVCMBS. Afterwards, the simulation model is explained in detail in Section 5. Section 6 defines the replication decision policies proposed for the directory service. Section 7 validates our approach in computational experiments, providing insights into the problem through the consideration of different directory service configuration and parameter settings. Finally, Section 8 summarizes the main contributions and results of this paper, and identifies lines of future research.

2 Literature review

Resource management is one of the crucial tasks in VC systems, which is due to the intermittent availability and heterogeneity of the nodes. This section focuses on works about the selection of resources in distributed large-scale systems based on heterogeneous and non-dedicated components. More detailed information can be found in [8], which provides an extended literature review of resource allocation in VC.

2.1 Resource allocation in VC systems

As resource allocation is a key function of VC systems, several authors have worked on this topic. In [9] the authors propose the Multi Criteria Biased Randomized (MCBR) method, a selection method for large-scale distributed systems that use unreliable nodes, which is based on a multicriteria optimization strategy. They evaluate their method in a microblogging social network formed by a large number of microservices hosted by volunteer nodes. In [10] the same authors propose a simheuristic algorithm [11] for the resource allocation problem considering stochastic node quality. They consider node quality as a random variable which follows a log normal probability distribution function. This simheuristic approach generates allocation mappings that achieve a minimum level of reliability (which is an input parameter). A similar work was proposed in [12], where the authors propose SaRa, a probabilistic system for estimating the reliability of untrusted edge resources in volunteer cloud computing. Another interesting method for predicting the reliability in volunteer clouds is presented in [13], which proposes a method for predicting the availability and the reliability of the nodes of a volunteer cloud. The method is based on a multi-state discrete semi-Markov prediction algorithm. The proposed method

is validated in a real volunteer cloud system called cuCloud. In a subsequent work [14], the same authors present three heuristics for placing virtual machines in Volunteer Clouds. To compute the reliability of the nodes the proposed heuristics are based on the multi-state semi-Markov Process presented in the previous work. Alsenani et al. [15] propose a reputation and resource-based reliability model called ReMot. ReMot is an intelligent machine learning based model that utilizes historical data of the tasks and host machines to extract their resource usage patterns. In [16] a service placement algorithm, called BASP, is proposed to place micro-cloud services in community networks. The algorithm uses the k-Means algorithm for clustering and a light weight bandwidth computation/estimation heuristic. In [17] the authors propose various heuristics to distribute jobs, while maximizing the throughput done by the nodes, without violating established money budget constraints. The heuristics are based on the price of electricity consumed by the nodes, considering its temporal variation over time, and the CPU time used. In [18, 19] a method is proposed to schedule scientific and data intensive workflows, in order to enhance the utilization of VC systems. The proposed method increases the percentage of workflows that meet the deadline, satisfying the QoS constraints in terms of the deadline, minimum CPU speed, and minimum RAM or hard disk requirements. The proposed workflow scheduling system partitions a workflow into sub-workflows, in order to minimize data dependencies among the sub-workflows. In contrast to previous work, we propose a class of probabilistic sort and select directory service policies for selecting resources in a DVCMBS and provide a Pareto analysis of all of the possible directory service configurations in terms of reliability and efficiency maximization.

2.2 Resource allocation in voluntary P2P systems

Node reliability is a key issue in Peer-to-Peer (P2P) systems. In P2P systems, Unlike in a DVCMBS, there is no central server that decides to allocate a job or data to a particular node, making resource allocation challenging. Ramachandran et al. [20] propose a P2P desktop grid framework for submiting computing jobs to non-dedicated resources. The framework utilizes a resource availability prediction method, using group availability data, to predict the probability that the nodes will be available at a given time. Graffi et al. [21] present $P^3R^3O.KOM$, a distributed mechanism to reserve, monitor and use P2P resources and which maintains long-term resource reservations through controlled redundant resource provision. Another interesting work is proposed in [22]. The authors propose a simple and effective meta-heuristic, inspired by behavior of bark beetles in nature, for search and lookup in an unstructured P2P system.

Regarding data availability and reliability in P2P storage systems, how to determine data redundancy is a key issue which has been widely studied during recent years. Tienan et al. [23] propose a replication model to improve the data availability of P2P storage platforms. The proposed replication model is based on semi-Markov chains which are used to estimate the probability of a peer's failure. Meng et al. [24] present a churn-aware durable P2P data storage scheme. The method is based on calculating the probabilities that peers are online in each time interval. It then groups the peers which have complementary online patterns creating so-called generalized peers which have high simultaneous online probability in each time interval. Xiao et al. [25] present a novel technique for calculating the probability of peer failure during a given period. Based on this technique, the method can evaluate the failure probability of any group of peers. This information is then used to estimate the number of replicas required to achieve better trade-offs between performance and data availability in P2P platforms. Another interesting approach is GLARAS [26], a locality aware replication method, which reduces the average access latency by grouping data requesters and allocates unique replicas for each subgroup. In a later work, the same authors, present Pyramid [27], a similar approach to GLARAS, but also considering availability-awareness of replicas. In P2P storage systems using erasure codes, data blocks are encoded in many redundancy fragments. These fragments are then sent to distinct peers of the network. Caron et al. [28] present three different placement policies, two of them are local, in which the data are stored in logical neighbors, and the other one, global, in which the data are spread randomly in the whole system. They study the impact of these placement policies on the probability of data block loss and the bandwidth consumption to maintain data redundancy. A similar work is presented in [29]. The authors propose a fragment placement strategy to optimize the usage of peer resources and ensure symmetry in the system. For node lookup and resource discovery in the P2P network the method uses an implementation of Kademlia which is a type of distributed hash table. Some other data replication policies for P2P applications have been reviewed in [30, 31].

2.3 Online social networks

Due to the increasing popularity of social networks, many works have studied the assignment of resources in Distributed Social Networks, which run over large-scale distributed systems. In [32] the authors design three heuristics for solving the client-

server assignment problem in online social network applications, maintaining a certain level of load balance. These algorithms are based on user communication patterns. In [33] the authors develop three heuristics to assign clients to servers in continuous Distributed Interactive Applications (DIAs) [34]. They reduce network latency to maximize the interactivity under consistency requirements. The minimum achievable interaction time for DIAs to preserve consistency is analyzed. A complementary study is presented in [35]. The authors put forward two efficient server placement algorithms for hosting continuous DIAs, which are designed to optimize interactivity performance. The algorithms take into account the interaction between clients, considering their path in the network and the latency. In [36] the authors propose a heuristic relying on relaxed convex optimization which, given a communication pattern among the clients, provides an approximately optimal client–server assignment for a pre-specified trade-off between load balance and communication. This heuristic can be used in distributed applications such as Instant Messaging Systems (IMS). The methods previously presented are based on profiling user behavior (*i.e.*, obtaining information about user communication patterns) in specific social networks. Our methods differ from these approaches because we only gather information of the network and the nodes (computing resources), making it unnecessary to analyze the user behavior to make the assignments. Moreover, our methods are developed to be applied to community-owned systems, which are composed of unreliable nodes.

3 Problem Description

In this work we address the problem of developing blog replica decision policies for the directory service of a distributed volunteer computing micro-blogging platform. In more detail, the directory service is the centralized component of a hybrid p2p infrastructure formed by peers voluntarily contributed by participants in the social network. As explained below, it is in charge of both deciding which peer will host each blog replica and resolving the allocation of each blog replica. Therefore, a distributed volunteer computing micro-blogging service (DVCMBS) provides a platform for a community of bloggers in which messages can be transmitted for all other bloggers to see. Within a volunteer computing micro-blogging service there is no central database for recording all previous messages. Instead, some bloggers of the volunteer computing community donate a portion of their computer's resources to the blogging service as storage nodes, indeed some nodes might not donate any storage capacity

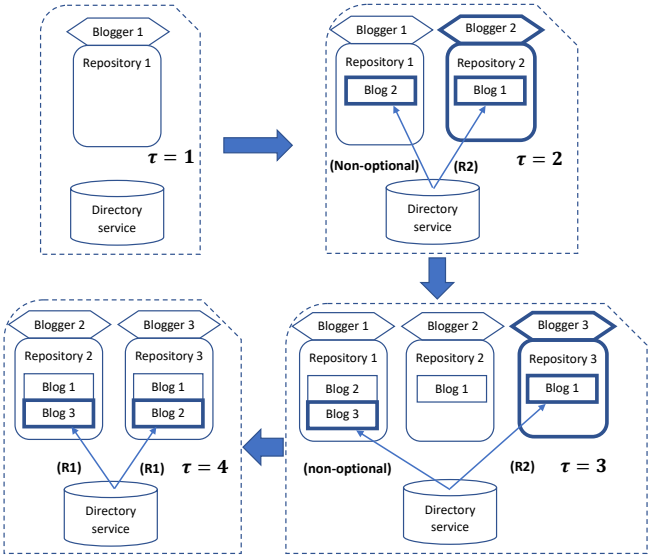

Figure 1: Diagram of possible replication decision epochs.

at all. These combined resources are used to create a parallel distributed database. The nodes of such a service are voluntary and are under no obligation to remain online to ensure that the full record of previous messages remains available. The challenge is then to develop a directory service which undertakes the role of deciding on which donated storage resources to store replicas of blogs. The goal of a directory service policy is to maximize the availability of all messages over time, with replica efficiency (in terms of the total number of replicas generated) as a secondary goal. Each combination of a blogger and a node is assigned a random ID, thus protecting the data of the blogger.

The directory service considers replication decisions at two different types of decision epoch: 1) when a computer arrives online, and 2) when a computer goes offline. I.e., we consider an event-based directory service.

For the case of 1), the new online computer may be a blogger signing in, in which case replicas of the new arrivals set-of-previous-messages (referred to as a *blog*) can be stored within the denoted storage space (referred to as a *repository*) of another online computer. This action is depicted by the left arrows in the second and third time intervals of Figure 1. Additionally, the new online computer may be a computer that donates a repository to the DVCMBS, in which case blog replicas can be stored within the repository of the new online computer. This action is depicted by the right arrows in the second and third time intervals of Figure 1. Which of these is actually considered depends on the configuration of the directory service (see Section 5 for details).

For the case of 2), a directory service may be able to identify valid online replicas of the replicas that were stored within

the repository of the leaver and proceed to make additional replicas of them. This case is depicted in time interval 4 of Figure 1. Which of the replication decision epochs, that are actually considered, depends on the configuration of the directory service (see Section 5 for details).

Directory service replication decision policies are required to ensure that the volunteer computing micro-blogging service functions reliably. In this work reliability is defined as the average rate at which all messages are available within the repositories of online nodes. We seek to maximize both reliability and replication efficiency.

4 A model for a volunteer computing micro-blogging service

In this section the details of the example volunteer computing micro-blogging service are presented, including any assumptions.

The DVCMBS is composed of computing nodes, namely personal computers and laptops. These include a node which hosts the directory service. The directory service is assumed to be the only computing node that is always available. In order to maintain full directory service availability 24x7, it may be replicated in a cluster. The remaining nodes are computers whose online presence is uncertain, since computers can leave and enter the system as frequently as they choose. Nodes can also donate any amount of computational resources they choose to, including none, to deploy and operate the system. A computer can correspond to: 1) a blogger who donates storage (not RAM) resources; or 2) a donated resource with no associated blogger. In reality, computers can be a community resource that can be used by any number of bloggers. Such community resource nodes can be modeled as a combination of nodes of types 1) and 2). Furthermore, a blogger may have access to multiple computers with some donating storage resources and others not. Such a blogger can also be modeled using a combination of nodes of type 1) and 2). In any case, the directory service is in charge of assigning blog replicas to the donated distributed storage capacity. Furthermore, it is assumed that the directory service has access to a record of the blog replicas stored in each repository. This assumption is required for two reasons. Firstly, the directory service needs to avoid allocating multiple replicas of the same blog to the same repository. Secondly, such a record can be used to determine which blogs are in most urgent need of additional replicas. Each computing node has a set of parameters governing its behavior. These include a node's arrival rate (if offline), a node's distribution of online duration, an associated messaging rate, the distribution of the sizes of the messages and the amount of storage that a node has donated to the DVCMBS. Nodes may have a zero messaging rate or donated storage amount. In this work both node online duration and message size [37] are assumed to follow log-normal distributions. This is appropriate because online duration and message size are always non-negative, it is also a very versatile distribution capable of modeling a wide variety of behaviors. We also note that this is not a structural requirement of the proposed method any other distributions can also be used. Nodes are each assumed to arrive online according to an exponential distribution. This assumption permits a concise theoretical presentation of our proposed approach. Javadi et al. (2009) [38] compare the goodness of fit for a range of statistical distributions for the availability and unavailability of volunteered computers. They find that the availability of 34% of hosts follows a truly random process. They find that a hyper-exponential distribution is often a good fit for node availability, which is a weighted sum of exponential distributions. The exponential distribution represents a worst case scenario in terms of the predictability of the availability of nodes. A policy geared towards this worst case scenario is suitable in a volunteer computing context since nodes are under no obligation to remain online. Our results can be viewed as lower bounds on performance levels for cases in which node availability is more predictable. We note that the proposed method works with any other distributions including empirical ones. Furthermore, different distributions can be used for individual nodes and for different times of day. Full details of the node parameters considered in this work are provided in Section 7.1.

This work assumes that each blogger keeps a separate copy of their own blog. This assumption means that if all replicas of a blog are lost whilst the associated blogger is offline, then complete replicas can be generated when the blogger arrives back online. Blog replica availability can be treated as a binary issue, either available (and complete) or unavailable. Binary blog replica availability simplifies the analysis, making the focus that of maintaining the availability of replicas of all blogs. The alternative case, where blogs can be lost forever, results in a more complex decision making problem. The objective would then be to maximize the total amount of retained replicated messages, making it beneficial to prioritize the replication of frequently updated blogs. This can lead to discrimination against bloggers who post few messages. The proposed approach can be considered a fair solution for this alternative case. In the case in which blogs can be lost forever, allocating a replica to a blogger's own repository provides an incentive for bloggers to leave their com-

Summary of the notation	
Directory service configuration options	
$R1$:	Directory service replication option 1. Determines whether or not the directory service attempts to generate replicas of leaver repositories, using valid copies stored by bloggers who are currently online.
$R2$:	Directory service replication option 2. Determines whether or not the directory service attempts to generate replicas of arriving blogger's blogs.
$V1$:	Directory service validation option 1. Determines whether or not the validity of an arriving blogger's repository entries are established using the directory service's record of blog entry counts.
$V2$:	Directory service validation option 2. Determines whether or not the validity of an arriving blogger's repository entries are established by comparison with valid copies stored by other online bloggers.
$V3$:	Directory service validation option 3. Determines whether or not the directory service attempts to replace outdated replicas with up-to-date copies.
$V4$:	Directory service validation option 4. Determines whether or not the directory service deletes blog replicas whose validity can not be immediately proven, or leaves them until their validity has been proven.
Replica notation	
m_{bt}:	The size of blog b in time interval t.
n^a:	The total number of times that replicas of blogs are available when required.
n^r:	The total number of times that blog replicas are required.
n^g:	The total number of replicas generated.
r:	Denotes a replica in general.
r^b:	Denotes a replica of blogger b's blog.
r_c^b:	Denotes a replica of blogger b's blog, stored in node c's repository.
Directory service replication policy parameters	
β:	Biased randomization geometric distribution shape parameter.
F_1:	Quantile parameter used for estimating online duration from the online duration distribution of a blogger.
F_2:	Quantile parameter used for estimating offline duration from the offline duration distribution of a blogger.
M:	Directory service replication policy parameter which specifies a maximum number of replicas that can be stored in an arrival's repository, immediately upon arrival (replication decision epoch 2).
N:	Directory service replication policy parameter which specifies a target number of replicas of each blog to maintain.
P:	Directory service replication policy parameter which specifies a target probability that at least one replica of each blog remains available within the set of online repositories.
Sets	
A:	The set of bloggers who have arrived in the DVCMBS in the previous time interval.
B:	The set of bloggers.
C:	Candidate list of host repositories, for storing a replica of a given blog.
D:	The set of blogs.
L:	The set of bloggers who have left the DVCMBS in the previous time interval.
R^b:	The set of online replicas of blogger b's blog at the current time.
R_t^b:	The set of online replicas of blogger b's blog at time interval t.
Other parameters	
$\delta(b)$:	The increase in the probability that at least one replica of blog b remains available online before blogger b returns back online after leaving once, due to the creation of an additional replica of blog b. (See Equation 5)
λ_b':	The estimated (Poisson process) arrival rate of node b, when node b is offline.
$p(r)$:	The probability that the host repository, of a replica r, remains online until the source blogger of replica r arrives back online, after leaving once.
q:	Used to denote that probability that a node is still online at a given time, given the time they arrived online.
$\rho(R^b)$:	The probability that at least one of the replicas of blogger b's blog remains online, before blogger A returns back online, after leaving once.
S:	The number of simulation repeats used to evaluate a given combination of directory service configuration options and replication policy parameters.
S_b:	Standard deviation parameter of node b's (log transformed) online duration distribution.
t_b:	The time at which node b last arrived online.
t_b':	The estimated time at which node b will arrive online next.

puters online, since this will increase the availability rate of their blog.

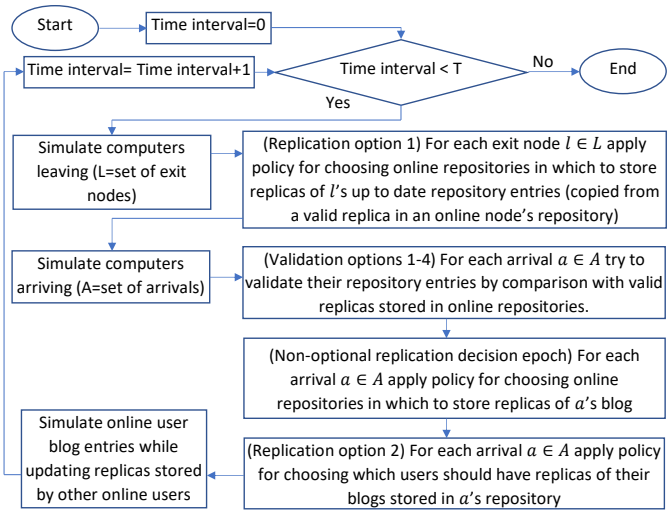

Figure 2: Flowchart of the simulation model for the DVCMBS.

5 Simulation

This section describes a simulation model of the volunteer computing micro-blogging platform that is considered in this work. The simulation model is both time discretized and event-based. In each time interval node arrival and exit events are simulated along with blogger messaging behavior. Figure 2 is a flowchart representation of the simulation model.

Figure 2 shows that within each simulated interval of time node exit events are simulated first. An exit event occurs when a randomly sampled online duration has been completed. The set of exit nodes for a given time interval is denoted L. The simulation then considers each exit node ($l \in L$) and considers the replication of the exit node's repository entries. This type of decision epoch requires that the directory service identify up-to-date replicas of leaver's repository entries that reside within the repositories of the remaining online computers. The replication of leaver repository entries is a true/false option in the simulation, referred to as *replication option 1* or ***R1*** for short. In Section 7.3 results are provided which indicate the impact of different simulation configurations.

Following the consideration of the set of exit events, node arrival events are simulated. The set of arrivals is denoted A. The repositories of new arrivals may contain repository entries that are still valid and up-to-date. Therefore, it can be beneficial, from a reliability perspective, to check the validity of the new arrival's repository entries. Valid entries can be retained, which in turn increases the probability that at least one replica of those entries remain available online. An additional and related option is the updating of outdated replicas, once they are proven to be outdated. For the case of a directory service configuration in which

no repository entry validation checks are performed, repositories can be deleted every time a computer leaves the DVCMBS. The set of Boolean *validation options* is enumerated as follows.

- ***V1****: Check arrivals' repository entry validity against directory service feed entry counts.* This repository entry validation approach relies on the assumption that the directory service keeps a record of the number of entries made in each blog.
- ***V2****: Check arrivals' repository validity against valid online replicas. This validation approach is more general than V1 as it does not require that the directory service maintain counts of the number of blog entries made by each blogger. V2 can be used instead of V1, but has no additional benefit if used in conjunction with V1.*
- ***V3****: Update out-of-date replicas.* This option refers to the possibility of updating outdated repository entries. This is achieved by a copying operation involving a different online valid copy. V3 is only applicable in cases where V1 or V2 is true.
- ***V4****: Retain repository entries until validity is proven.* For cases where the validity of repository entries cannot be proven, those repository entries may be retained until their validity can be proven or until a valid replica of the same blogger's feed is added to the same repository. V4 is only applicable if V3 is applicable.

Following the validation of repository entries, two types of arriving node related replication decisions can be considered. Firstly, replicas of the arrival's blog can be stored in the repositories of other online computers. This step simulates a blogger signing in and being assigned to repositories. The consideration of the replication of an arrivals blog is non-optional. Secondly, blog replicas stored within other online repositories can be stored within the repository of the new arrival. This optional replication decision is referred to as *replication option 2* or ***R2*** for short. Following this step the simulation simulates the messaging behavior of all online bloggers. Every time a blogger posts a new message the directory service updates the replicas of that blogger's blog. The simulation continues until a specified number of time intervals have been completed. In Section 7 we consider, among other things, the impact of different sets of choices for replication decisions *R1* and *R2* and replica validity checking options *V1*-*V4* in their effect on the overall replica availability and replication efficiency. In instance of a set of choices for options R1, R2, V1, V2, V3 and V4 is referred to as a directory service configuration.

6 Decision policies for the directory service

In this section we describe the general class of directory service decision policies that are considered in the remainder of this work, namely sort and select based policies. A sort and select approach creates a list of the available decisions sorted in decreasing order of a decision score. A decision is then selected from the list with higher ranked decisions being assigned a higher probability of selection. Such policies are applied whenever we need to choose a repository in which to store a given blog replica or to choose a blog to replicate and store within a given repository. In this case we sort the candidate repositories (or blogs) according to a measure of reliability of each replication decision. Replication decisions are then made sequentially in decreasing reliability order until the termination criteria is reached. The sorting criteria determines *which* replicas are made (see Section 6.2), while the termination criteria determine *how many* (see Section 6.3) replicas are made in each decision epoch. In the remainder of this section we specify the measures of reliability that are used to sort candidate repositories (blogs) in each replication decision epoch.

6.1 Sorting policies for different replication decision epochs

In general, the directory service can be faced with three different types of blog replication decision epochs, which are enumerated as follows.

- *Replication decision epoch 1:* A computer arrives online and replicas of the blog (b) associated with a blogger using that computer can be stored within the repositories of other online computers. In this work candidate repositories ($c \in C$) are sorted in *decreasing order of the probability* $p\left(r_c^b\right)$ (Equation 3). $p\left(r_c^b\right)$ is the probability that node c remains online before node b returns back online after leaving once. r_c^b denotes a replica of node b's blog stored in the repository of node c. This criterion reflects this work's focus on ensuring the availability of replicas of blogs whilst the associated bloggers are offline, since the directory service can allocate new replicas once the blogger arrives back online. Section 6.4 outlines how this probability can be estimated based on the prior observed behavior of each computing node. Figure 3 outlines the process followed in replication decision epoch 1.
- *Replication decision epoch 2:* A computer arrives online

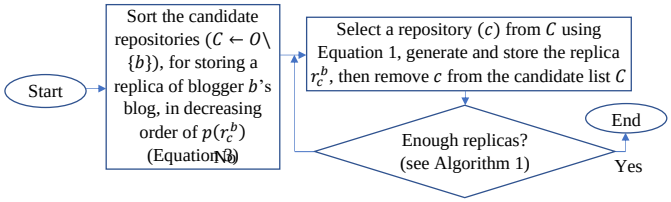

Figure 3: Flowchart of the process of selecting repositories for storing replicas of blogger b's blog.

and donates repository space. The decision in this case is which, if any, additional blog replicas should be stored within the new available repository space. The candidate blogs ($b \in D$) for replication and storage on the new available repository space, are sorted in *decreasing order of* $\delta\left(R^b\right)$ (Equation 5). $\delta\left(R^b\right)$ is the increase in the probability that at least one replica of blog b remains available online, before the blogger b returns back online after leaving once. Where R^b denotes the current set of available replicas of blog b. Section 6.4 specifies how the probability that at least one replica of a given blog can be calculated. The calculation is based on the estimated probabilities that each replica remains available online before the blogger returns back online after leaving once.

- *Replication decision epoch 3:* A computer goes offline and the directory service considers the generation of additional replicas of the blog replicas that were stored within the repository of the leaver. This decision epoch relies on the assumption that the directory service has access to a record of the contents of all repositories, including the offline repositories. The decision in this case is which repository entries should be replicated and in which repositories they should be stored. A candidate repository entry is feasible for replication if a valid replica exists within the repository of a different online computer. In this work, candidate leaver repository entries $w \in W$ are sorted in *increasing order of the probability* $\rho\left(R^{blogger(w)}\right)$, see Equation 4, where $blogger(w)$ denotes source blogger of replica w. $\rho\left(R^{blogger(w)}\right)$ is the probability that at least one of the remaining replicas remains available online until after $blogger(w)$ returns back online. The directory service then considers the exit node's repository entries in this order, and for each applies the criteria outlined in *replication decision epoch 1* (above) to determine on which repositories additional replicas are stored.

6.2 Which replica?

Having sorted the candidate repositories for storing a replica of a blog, single candidates are selected sequentially until some pre-specified termination criteria are satisfied (see Section 6.3 for the details of the termination criteria applied). We consider three possible approaches for selecting replication decisions from sorted candidate lists: 1) greedy; 2) random; and 3) biased randomized. For the case of greedy selection, we simply take the candidate from the top of the sorted list. For the case of random selection, a candidate is selected from the candidate list according to a uniform random distribution. For the case of biased randomized selection [7], candidates are selected from the candidate list according to a geometric distribution. Given a uniform random input *uniRand* in the interval $[0,1]$ a geometrically distributed random candidate list position index is given by the following equation:

$$index = Mod\left(\left\lfloor \frac{\ln\left(uniRand\,(0,1)\right)}{\ln\left(1-\beta\right)} \right\rfloor, |C|\right), \qquad (1)$$

where β is the parameter of the geometric distribution that controls the skew of the geometric distribution. β close to 0 models random candidate selection, $\beta = 1$ simulates greedy candidate selection and $0 < \beta < 1$ produces greedy but randomized candidate selection. In the experiment results (Section 7), we compare the effect of using $\beta \in \{0, 0.5, 1\}$ for candidate selection and consider their effect of blog replica availability and the total number of replicas generated per blog. Since Equation 1 is undefined for $\beta = 0$, we use $index = \lfloor uniRand\,(0,1)\,|C| \rfloor$ to model a purely random selection.

6.3 How many replicas?

Each time additional replicas of a blog are being made (see blog replication decisions 1) and 3) in Section 6.1) the criteria outlined in Section 6.1 are used to sort the available host repositories. The number of replicas generated is determined by two parameters: *i*) the target number of replicas per blogger (N) and *ii*) the target probability (P) that at least one of the replicas of a blog remains available online before the blogger, whose blog is being replicated, arrives back online after leaving once. The termination criteria is expressed in pseudocode in Algorithm 1.

Algorithm 1 defines the termination criteria for generating replicas of a given blog b. In general, the number of replicas generated before these termination criteria are satisfied increases with N and P. P differs from N as a termination criteria parameter because achieving the minimum P value depends on the behaviors associated with the particular repositories selected for

Algorithm 1 Termination criteria when generating replicas of blog b

1: Inputs: C sorted candidate list of available host repositories, R^b current set of available replicas of blog b
2: **while** $\left(|R^b| < N\right) \cap \left(\rho^b < P\right) \cap \left(|C| > 0\right)$ **do**
3: Select a host repository (c) for an additional replica (r) of blog b, using either random, greedy or biased randomized selection policy.
4: $R^b \leftarrow R^b \cup \{r\}$
5: $C \leftarrow C \setminus \{c\}$
6: Update ρ^b using Equation 4
7: **end while**
8: Output: R^b

hosting blog replicas. In this sense, N represents the simplest possible termination criterion. In Section 7 we present results which indicate the influence of different pairs of values for N and P on blog replica availability and the total number of replicas generated per blog.

For the case of blog replication decision 2), the maximum number of blog replicas that can be stored within the repository of an arrival node is an input parameter denoted M. This parameter is designed to prevent the overloading of the computational resources donated by the arrival node when they arrive. In Section 7 we present experimental results which indicate the influence of the value of M on blog replica availability and the number of replicas generated per blog.

6.4 How to estimate probabilities that replicas remain available whilst they are required

This section presents a method for calculating the probability that blog replicas remain available until the associated blogger arrives back online after leaving once. We assume that computing nodes arrive online according to a Poisson arrival pattern and that the online duration of computing nodes follow a log-normal distribution. Whilst the following methods are based on these assumptions, we note that equivalent calculations exist for other distributions including empirical ones.

In order to calculate the probability ($p(r)$) that a replica r of blog a, hosted by the repository of node b, remains available online before the expected time ($t_a^{'}$) that node a arrives back online after leaving the DVCMBS once, we firstly use the following equation to estimate $t_a^{'}$.

$$t_a^{'} = t_a + \left(e^{\bar{x}_a + \sqrt{2(s_a)^2}\,\mathrm{erf}^{-1}(2F_1 - 1)}\right) + \left(-\frac{1}{\lambda_a^{'}} ln\left(1 - F_2\right)\right). \quad (2)$$

The second term of Equation (2) is the quantile function (F^{-1}) of the log-normal distribution for sampling online duration times of bloggers. The third term is the quantile function for sampling exponential distributed offline duration times. t_a is the time that node a last arrived online; $\bar{x}_a$ is the estimate of the mean parameter of node a's (log transformed) online duration distribution; s_a is the standard deviation of node a's (log transformed) online duration distribution; $\lambda_a^{'}$ is the estimated arrival rate of node a; F_1 is the quantile parameter for the log-normal quantile function (second term) that is used to calculate a node's estimated expected online duration; and F_2 is the quantile parameter for the inter-arrival time of a Poisson arrival process (third term). For the case where $F_1 = 0.5$ we obtain the expected online duration of node a, however $F_1 > 0.5$ can be used to generate an over estimate, whilst $F_1 < 0.5$ can be used to generate an underestimate estimate. F_2 can be set equal to F_1 and for the same reason. In this work we consider the case $F_1 = F_2\,(= F)$. Following this, the cumulative density function of the log-normal distribution (Equation 3) can be used to estimate ($p(r)$) the probability that a replica r of blog a stored in node b's repository remains online until node a returns back online at time $t_a^{'}$.

$$p(r) = 1 - \left(0.5 + 0.5\mathrm{erf}\left(\frac{\ln\left(t_a^{'} - t_b\right) - \bar{x}_b}{\sqrt{2}s_b}\right)\right), \quad (3)$$

where erf is the error function and $t_a^{'} - t_b$ is the expected amount of time after node b arrived online that node a is expected to arrive back online after leaving once. Equations 2 and 3 are based on the exponential distribution and log-normal distribution equations in [39].

Equation 3 is the sorting criteria for *Replication decision epoch 1* (Section 6.1). The reasoning behind using $p(.)$ as a criteria for sorting candidate repositories, for storing a blog replica, is that it directly measures the probability that the replica will ensure the online availability of a blog whilst the associated blogger is offline. If replicas of a blog become unavailable while the associated blogger is offline, the directory service cannot allocate new replicas. This approach accounts for the last observed online arrival times of nodes a and b and also their estimated online and offline duration distributions.

Given a set of replicas ($r \in R^a$) of blog a, each with respective estimates of $p(r)$, the probability $\rho(R^a)$ that at least one of these replicas remains available online before the corresponding blogger arrives back online after leaving once. This can be estimated as one minus the probability that none remain available, as follows:

$$\rho\left(R^a\right) = 1 - \prod_{r \in R^a} \left(1 - p\left(r\right)\right). \quad (4)$$

Equation 4 can be used to calculate the impact of storing an addition replica of a blog in a particular online repository. In particular, for the case of *Replication decision epoch 3* (Section 6.1), where we consider which of an exit node's repository entries are in most need of additional replicas, Equation 4 is used for sorting the repository entries.

For the case of *Replication decision epoch 2* (Section 6.1), where we consider which blog should be replicated and stored within the repository of an arrival node's repository, we use the following equation to sort the candidate blogs $d \in D$.

$$\delta\left(d\right) = \rho\left(R^d \cup \left\{r'\right\}\right) - \rho\left(R^d\right), \quad (5)$$

where $\delta\left(d\right)$ is the increase in the probability that at least one replica of blog d remains available online before blogger d returns back online after leaving once. r' denotes the candidate new replica of blog d.

Each of the calculations outlined above rely on maintaining estimates of the expected times that each computing node will arrive back online after leaving once. To do this, Equation 2 is applied every time a computing node arrives back online. We also note that it is necessary to update the estimated next-arrival-time when a node has not left by the time it is expected to leave (given by the sum of the first two terms of Equation 2). In such a case, the updated next-arrival-time estimate is based on sampling the tail of the online duration distribution starting from the current time. In particular, the log-normal distribution cumulative density function (which is 1-Equation 3) is used to find the start of the tail region of the node's online duration, which is the probability (q) that the node is still online given the time they arrived online. Following this, Equation 2 is applied with $q + (1-q)F_1$ in place of F_1 to update the estimated next-arrival-time of the node.

7 Experiments

This section analyzes the trade-off between reliability and efficiency of DVCMBS directory services operating in all possible configurations $(R1, R2, V1, V2, V3, V4)$ and for a range of replication decision parameters (N, M, P, F, β). We firstly describe the details of the tests instances and the measures of reliability (average blog replica availability rate) and efficiency (average number of replicas generated per blog).

7.1 Description of instances

Due to the lack of benchmark test instances for the problem considered, the experiment results presented in this section are based on a test instance defined by the following:

- 100 time intervals, with each time interval representing a duration of one unit of time. (This work focuses on the finite time horizon case.)
- 50 bloggers/computers. (This relatively small number allows us to perform an exhaustive investigation of many possible directory service configuration and replication policy parameter combinations. In experiments involving larger numbers of bloggers, the results were qualitatively the same except that availability rates are improved. This is because of the associated increase in the number of nodes with very high online duration times.)
- Repository sizes of 1 Gigabyte. (Effectively infinite since we are not considering the storage constrained case.)
- Computer/blogger Poisson process arrival rates uniformly distributed in the range $[0.01, 0.4]$ per time interval. (Capturing both, infrequent and frequent, arrivals.)
- Log-normally distributed computer/blogger online sessions with mean duration uniformly distributed in the range $[1, 10]$ per time interval and standard deviations uniformly distributed in the range of $[1, 10]$. (Capturing a population with a mixture of short and long online sessions.)
- Bloggers with blog entry rates uniformly distributed in the range $[0.1, 0.9]$ per time interval. (Capturing infrequent and frequent messaging behavior.)
- Log-normally distributed blog entry sizes with mean sizes uniformly distributed in the range $[1, 100]$ bytes and standard deviations uniformly distributed in the range $[1, 20]$ bytes. (Capturing short and long message behaviors each with low and high message size variance. We note that the proposed approach can be extended for the case where message size distributions are multi-modal, due to the inclusion of messages containing pictures and/or video files.)

The rationale behind these choices of node behavior parameters is to create an instance that is fully heterogeneous with respect to all of the behavioral parameters. This approach is designed to ensure that the problem instance is difficult in terms of not having any pathological features which may be directly exploitable. Additionally, the instance may be considered small,

this approach is designed to enable us to perform a full factorial investigation of directory service configurations and a fairly exhaustive investigation of the replication policy decision parameters.

7.2 Analysis of results

Replica availability and replica efficiency are the main performance measures with which alternative blog replication policies are to be assessed. The overall blog replica availability rate is calculated using the following equation:

$$\frac{n^a}{n^r}, \tag{6}$$

where n^a is the total number of times that replicas of blogs are available when they are required and n^r is the total number of times that replicas of blogs are required. A replica of a blog (b) is required in each time interval (t) for which a blog has a size greater than zero ($m_{bt} > 0$).

As a measure of blog replica inefficiency we calculate the average number of blog replicas generated per blog using the equation $\frac{n^g}{|B|S}$, where n^g is the total number of replicas generated in S repeat simulations and B is the set of bloggers. n^g also counts the act of updating an out-of-date replicas as a new replica since updating out-of-date replicas has an associated energy cost (assumed equal to be equal to the generation of a new replica).

$$n^r = \sum_{b \in B} \sum_{t=1}^{T} \begin{cases} 1 & if m_{bt} > 0 \\ 0 & otherwise \end{cases} \tag{7}$$

$$n^a = \sum_{b \in B} \sum_{t=1}^{T} \begin{cases} 1 & if |R_t^b| > 0 \\ 0 & otherwise \end{cases} \tag{8}$$

Our experimental results will be based on the simulation testing of a variety of candidate blog replication policies. Firstly, we identify the necessary number of repeat simulations in order for the resulting average blog replica availability rates to be representative.

Figure 4 shows the convergence of the average blog replica availability starting from different random seeds with respect to the number of repeat simulations. The results suggest that at least 100 repeat simulations are required for an average blog replica availability estimate error no greater than around 0.005. Based on this we have selected 300 repeat simulations ($log_{10}(300) = 2.48$) as the basis for the following results, since a tight level of convergence is achieved for this number of repeat simulations.

Hereafter, we adopt the convention of referring to *average blog replica availability rate* as ***availability*** and *average number of blog replicas generated per blog* as ***inefficiency***.

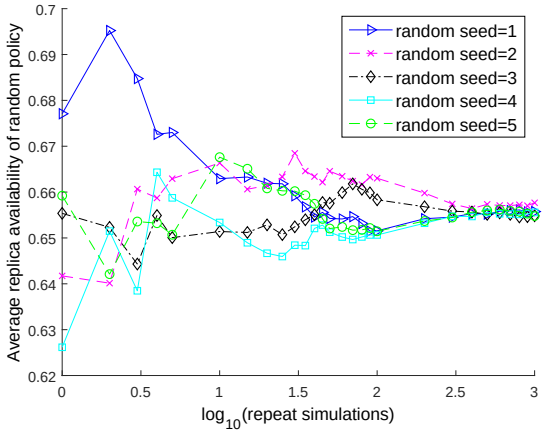

Figure 4: Convergence of the average blog replica availability starting from different random number seeds.

7.3 Full factorial experiment of directory configurations in conjunction with a wide sample of replication policy parameters

This section presents the results of simulation experiments of all combinations of directory service configuration options, namely $(R1, R2, V1, V2, V3, V4) \in \{true, false\}$, each implemented in conjunction with a wide sample of possible replication policy parameters, namely ($N \in \{1,2,4\}$, $M \in \{1,2,4\}$,$P \in \{0, 0.5, 1\}$,$F \in \{0, 0.5, 1\}$,$\beta \in \{0, 0.5, 1\}$). Each combination of directory service configuration options and replication policy parameters is tested in the problem instance described in Section 7.1 with 300 repeat simulations for each. In the following we analyze availability and inefficiency—the performance measures specified in Section 7.2. We begin by presenting and discussing the average availability rates and inefficiency measures associated with each choice of directory service configuration option and replication policy parameter. That is, for example, out of the 15552 experiments 7776, *i.e.*, half of those involve $R1 = true$, and of those experiments the availability rate was 0.936 whilst the inefficiency measure was 151.326, as shown in Table 1.

Table 1 provides an indication of the importance of each directory service configuration option as well as initial evidence regarding sensible parameter choices. Table 1 shows that by comparing the difference between the minimum and maximum average availability rates, associated with each choice for each input parameter, the most important parameter was P, since it has the largest gap between the minimum and maximum average availability rates as well as the highest average availability rate–associated with $P = 1$. Following P, the next most important parameters and associated parameter values were $N = 4$, $R1 = true$, $V2 = true$, $F = 0.5$, $V1 = true$, $R2 = true$, $V3 = true$, $\beta = 0$,

	Input decision	Parameter description	Treatment level (availability (inefficiency))		
			Low	Medium	High
directory service configuration options	R1 $\{false,true\}$	Replicate leaver's repository entries	0.856 (80.844)	-	0.936 (151.326)
	R2 $\{false,true\}$	Store replicas in arrivals repository	0.885 (114.860)	-	0.907 (117.310)
	V1 $\{false,true\}$	Validate replicas using message counts	0.880 (126.348)	-	0.912 (105.822)
	V2 $\{false,true\}$	Validate replicas using online replicas	0.874 (139.813)	-	0.918 (92.357)
	V3 $\{false,true\}$	Update out-of-date replicas	0.889 (108.539)	-	0.903 (123.631)
	V4 $\{false,true\}$	Retain replicas until validity proven	0.893 (116.141)	-	0.899 (116.029)
Replication policy parameters	N $\{1,2,4\}$	Target replicas per blog	0.800 (106.187)	0.933 (116.477)	0.956 (125.591)
	M $\{1,2,4\}$	Max replicas stored in arrivals repository	0.892 (115.212)	0.896 (115.940)	0.900 (117.103)
	P $\{0,0.5,1\}$	Target probability that one replica remains	0.754 (33.710)	0.959 (126.712)	0.975 (187.833)
	F $\{0,0.5,1\}$	Return time estimation quantile parameter	0.900 (71.068)	0.912 (112.491)	0.876 (164.696)
	β $\{0,0.5,1\}$	Biased randomization parameter	0.901 (106.214)	0.895 (119.379)	0.893 (122.661)

Table 1: Directory service options impact on average replica availability and inefficiency.

$M = 4$ and $V4 = true$ in that order.

In summary, based on the average availability rates, it appears to be beneficial to set all of the directory service configuration options to $true$, this makes sense since additional replication opportunities ($R1$ and $R2$) and replica validation and recycling opportunities ($V1$-$V4$) should all help to improve average availability rates. However, some of these options, also increase inefficiency. For instance, $R1 = true$, $R2 = true$ and $V3 = true$ each have the effect of increasing availability, which makes sense given that each of these options create additional opportunities for replica generation. On the other hand, $V1 = true$, $V2 = true$ and $V4 = true$ decrease inefficiency, which also makes sense given that these options increase the number of opportunities for reusing replicas from previous node sessions by performing validation checks on repository entries when nodes arrive back online. In regard to the replication policy parameters, with respect to availability, it was found to be beneficial on average to select relatively high parameter values for N, M and P, which makes sense since these are all replication decision termination criteria parameters which directly lead to the generation of more replicas. However higher values of these parameters also increased inefficiency. $F = 0.5$ was found, on average, to lead to the highest average availability rate, but only marginally better than $F = 0$ and $F = 1$. Low values of F also appear to decrease inefficiency. The choice of β (balance between greedy and random candidate selection) had a similar effect to that of F except that $\beta = 0$ (random candidate selection), on average, was found to provide both the highest associated average availability rates and lowest inefficiency. Random selection is a reasonable heuristic in the current context since it is conceivable that selecting a diverse range of host repositories for blog replicas may improve availability under uncertainty, by avoiding too much reliance on a few "reliable" nodes.

We now consider the trade-off between availability and inefficiency in more detail.

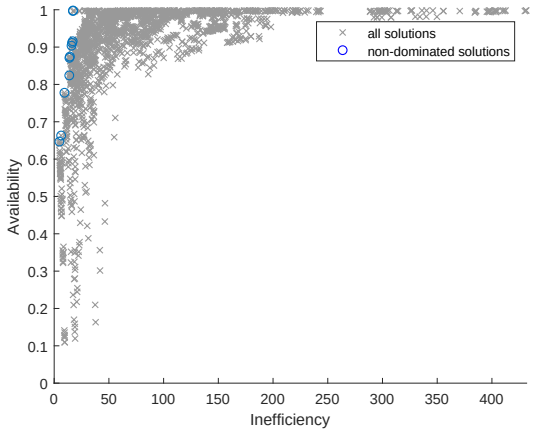

Figure 5: Availability and inefficiency measures for each combination of directory service configuration and replication policy parameters.

Figure 5 is a scatter plot of the average availability rates and inefficiency measures from each of the 15552 experiments, with the non-dominated experiment results displayed using circles. Figure 5 shows that it is generally the case, as expected, that high inefficiency is required for higher availability rates. It also shows that it was possible to attain availability rates close to 1, which also had moderate inefficiency. It was found that 11 of the 15552 experiments formed a non-dominated set of directory service policy parameters. We now take a closer look at details regarding those 11 non-dominated directory service configurations.

Table 2 provides the input parameters (both directory service configuration options and replication policy parameters), availability rates, inefficiency measures as well as the average number of available replicas per blog sorted in decreasing availability rate. Table 2 shows that the highest availability rates require the selection of all of the directory service configuration options ex-

Non-dominated solutions													
Input parameters											Performance measures		
R1	R2	V1	V2	V3	V4	N	M	P	F	β	Availability	Inefficiency	Average available replicas per blog
1	1	1	1	0	1	2	4	0	0	0.5	0.997	17.217	2.776
1	0	1	1	0	1	2	2	0	0	0.5	0.996	17.161	2.772
0	1	1	1	0	1	2	4	0	0	1	0.915	16.909	2.410
0	1	1	1	0	1	2	2	0	0	0.5	0.911	16.292	2.368
0	1	1	1	0	1	2	2	0	0	1	0.903	15.853	2.329
0	1	1	1	0	1	2	1	0	0	0.5	0.874	14.358	2.138
0	1	1	1	0	1	2	1	0	0	1	0.871	13.913	2.120
0	1	0	1	0	1	2	1	0	0	1	0.824	13.789	2.070
0	0	1	1	0	1	2	1	0	0	1	0.777	9.352	1.614
0	0	1	1	1	0	1	1	0	0	1	0.663	6.361	1.155
0	0	1	1	0	1	1	1	0	0	1	0.647	4.799	1.053

Table 2: Directory service options impact on availability, inefficiency and average available replicas per blog.

cept for that of $V3$, with a moderate choice of N ($N = 2$ rather than $N = 1$ or $N = 4$) and a low choice of P. It appears that high values of M are essential if the highest availability rates are to be achieved, since M is approximately decreasing with availability rate within the non-dominated set of directory service policy parameters. This makes sense considering that M is a replication termination criteria for replication decision epoch $R2$, which considers which blogs require replication and storage within the repository of an arrival node. Such arrival nodes have an increased chance of remaining online while other bloggers are offline, due to them having the most recent arrival times. The consideration of replication decision epoch $R2$ was found to be vital for achieving the highest possible availability rates. It was also the case that each of the non-dominated directory service policy parameter sets involved $P = 0$ and $F = 0$. As discussed above, low values of P and F were found to lead to low inefficiency. What is surprising is that the non-dominated set of directory service policy parameters includes those with high availability rates for $P = 0$ when it was found (Table 1) that on average P is increasing with availability rate. The explanation is that increasing P drastically increases the number of replicas generated per blog, which decreases the likelihood that experiments with high P values feature in the non-dominated set. Later on, Figure 9 shows that the particular value of F used has little effect on replica availability and efficiency, which is because it has only a small effect on the orders of the candidate lists used for each replication decision. It is also noteworthy that each of the non-dominated directory service policy parameter sets involved $\beta = 0.5$ or $\beta = 1$, whilst in Table 1 $\beta = 0$ provided, on average, the highest average availability rate. We must conclude that random candidate selection is beneficial in a large number of the perhaps nonsensical directory service policy parameter sets that were in the 15552 experiments. Within the non-dominated set of directory service policy parameter sets, it was found that a balance between greedy and random candidate selection is beneficial, which is achieved with $\beta = 0.5$. Lastly, the fact that greedy or biased greedy candidate selection ($\beta = 1$ or $\beta = 0.5$) are ubiquitous within the set of non-dominated directory service policy parameter sets can be taken as evidence supporting the effectiveness of the probabilistic candidate sorting criteria that were proposed in Section 6.1.

If there are operational constraints on the minimum average availability rates and maximum inefficiency, Table 2 can be used to identify the recommended directory service configuration and replication policy parameters. For instance, if we are tasked with guaranteeing a minimum average availability rate of 0.9 with the requirement that inefficiency is minimized, then we would select the fifth non-dominated directory service policy parameter set in Table 2. The experiments of Tables 1, 2 and Figure 5 have been repeated for the case where offline duration follows a log-normal distribution and online duration follows a Gamma distribution. [38] found that these distributions also often provide good statistical fits for the behavior of nodes in volunteer computing. The mean and variances were selected to match those of the previous experiment. The result are provided in an appendix and are qualitatively very similar to the results provided above.

In the following we consider the effects of varying the values of each of N, M, P, F and β independently in each of the non-dominated directory service policy parameter sets reported in Table 2. The reason for doing this is two-fold: firstly to see if further improvements upon the non-dominated directory service policy parameter sets are possible; and secondly to gain insights regarding the effect of each parameter in the important (non-dominated) directory service configurations.

Figure 6 (top) shows the effect of varying the value of N in

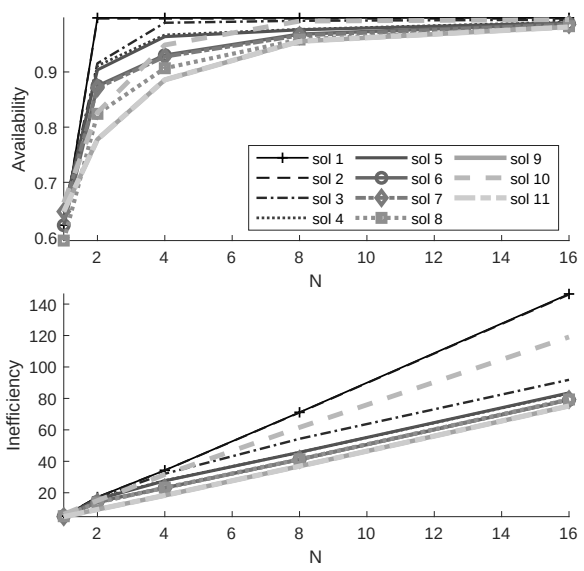

Figure 6: The effect of varying N on availability and inefficiency in each non-dominated directory service policy parameter set.

each of the non-dominated directory service policy parameter sets in Table 2. In each case availability increases with N whilst at the same time inefficiency increases linearly (Figure 6 (bottom)). N has a clear and predictable effect on availability and inefficiency. Non-dominated directory service policy parameter set 1 has the highest rate of increase in availability, this is because that directory service configuration considers all possible replication decision epochs as well as considering the updating of out-of-date replicas.

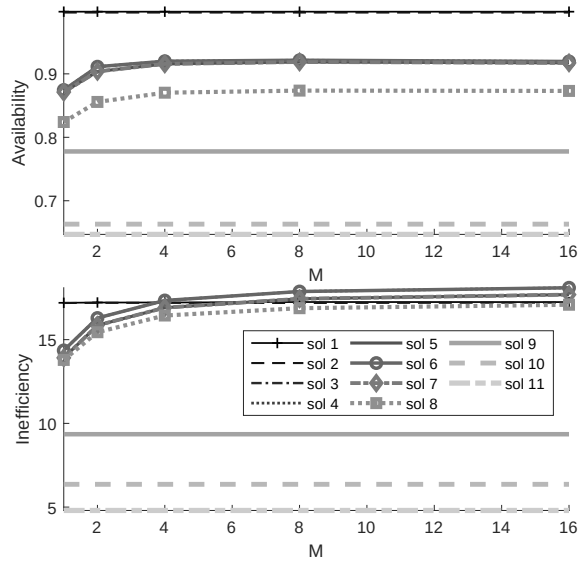

Figure 7: The effect of varying M on availability and inefficiency in each non-dominated directory service policy parameter set.

Figure 7 shows that increases in the value of M produce diminishing increases in availability rates. The reason for this is that each time the $R2$ replication decision epoch is entered, up to M additional blog replicas can be stored in the arrival's repository. However, in order to select which blogs are replicated and stored in the arrival's repository the non-optional replication decision epoch is entered to select them, the termination criteria of this decision epoch is parameterized by N. So if sufficient replicas of each blog already exist no more will be generated, regardless of the value of M. Figure 7 also shows that 4 of the non-dominated directory service policy parameter sets are insensitive to the value of M, this is because $R2$ is false in 4 of the non-dominated directory service policy parameter sets.

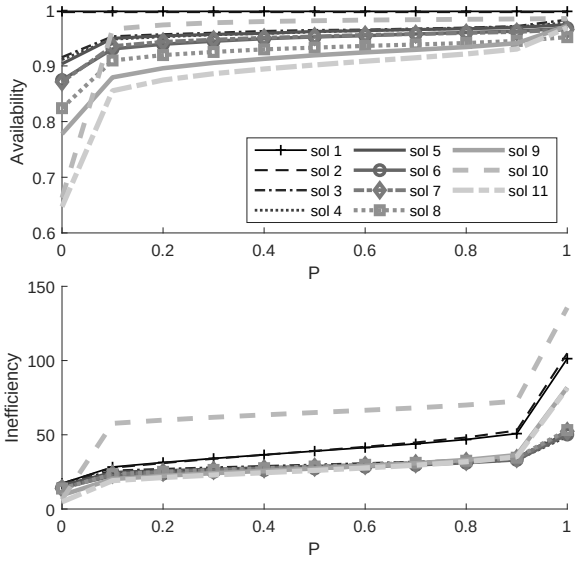

Figure 8: The effect of varying P on availability and inefficiency in each non-dominated directory service policy parameter set.

Figure 8 shows that increasing the value of P in each of the non-dominated directory service policy parameter sets increases both availability and inefficiency. This confirms the observations made earlier in reference to Table 1.

Figure 9 shows the effect of increasing the value of F in each of the non-dominated directory service policy parameter sets. In particular, it shows that availability and inefficiency are both relatively insensitive to the value of F except for the case of $F = 1$, which is because $F = 1$ results in predicted return times equal to infinity. This in turn results in predicted probabilities that replicas remain available online until the associated bloggers return of 0 in all cases, thus nullifying the probabilistic reasoning behind the ordering of the candidate lists in each replication decision epoch. The reason for the insensitivity of availability and inefficiency to the value of F is that the ordering of the candidate lists should be very similar in each case. This is because

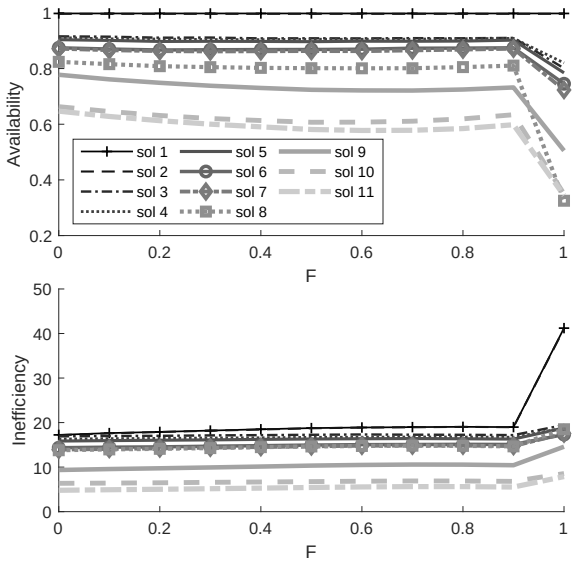

Figure 9: The effect of varying F on availability and inefficiency in each non-dominated directory service policy parameter set.

increasing (lowering) F decreases (increases) all estimated online and offline duration times, which are the basis of estimated probabilities that candidate replicas remain available. As a result, this leaves the order of candidate lists and therefore the selected candidates unaffected.

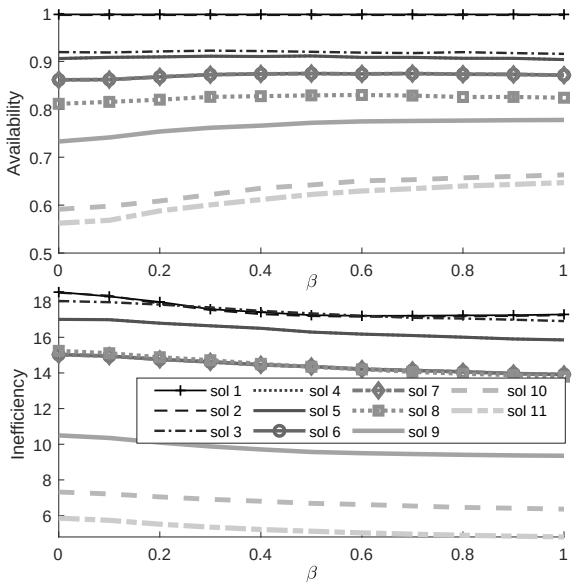

Figure 10: The effect of varying β on availability and inefficiency in each non-dominated directory service policy parameter set.

Figure 10 shows the effect of increasing the value of β in each of the non-dominated directory service policy parameter sets, and supports the earlier observation values of $\beta = 0.5$ and $\beta = 1$ are optimal. In particular, Figure 10 shows that increasing β improves both availability and efficiency. This is additional evidence supporting the idea that the probabilistic criteria used for sorting candidate lists (Section 6.1) are effective in regard to improving availability rates and the inefficiency required to achieve such rates.

8 Conclusions and future work

In this work we have considered the problem of developing blog replica decision policies for the directory service of a volunteer computing micro-blogging service (DVCMBS) with a focus on maximizing content availability as well as maximizing efficiency in terms of the number of replicas generated. As a basis for this investigation a discrete time and event-based simulation model of a DVCMBS was proposed and the possible blog replication decisions were enumerated. This work proposed a sort and select approach for determining which repositories are selected to host replicas of given blogs and for selecting which blogs require additional replicas given newly available repository space. Probabilistic sorting criteria, based on the probability that blog replicas remain available until a blogger returns online after leaving once, were proposed and equations for these were provided based on assumed distributions for node online and offline duration. Termination criteria were proposed for determining how many replicas of each blog are necessary.

A full factorial experiment of directory service configurations and replicas policy parameters was performed. The experiment results showed that high availability rates require large numbers of blog replicas. However, reasonable availability rates could still be achieved using directory service configurations that were quite efficient. In particular, it was found that utilizing all of the opportunities for blog replication and for recycling blog replicas stored in repositories in previous node sessions was essential for ensuring the highest availability rates. Of these it was found that the replication decision epoch $R1$, in which exit node repository entries are considered for replication, was the very important for achieving the highest availability rates. In regard to the parameters of the blog replication termination criteria, it was found that setting a minimum number of replicas per blog was an effective parameter for improving blog replica availability and that the number of blog replicas generated increased linearly with this parameter. As a result, moderate choices of this parameter were required for achieving a good trade-off between availability and efficiency. Another important parameter was the greedy randomness parameter of the biased randomization approach used for selecting candidate repositories and blogs for replication. In particular, each of the non-dominated directory service policy pa-

rameter sets involved either fully greedy candidate selection or a balance between random and greedy selection. This provided evidence that the proposed probabilistic sorting criteria are an effective mechanism for encouraging high availability and efficiency.

In future work we see several possible avenues to explore. Firstly, this work focused on the case where all computing nodes donate repository space, future work could consider the case whereby only very few computing nodes donate repository space. Secondly, although in this work the donated repository of each computing node was finite, the repository space of any nodes was never fully utilized. Therefore, future work could consider the case where some nodes donate small amounts of repository space. In such cases it may become necessary to replace the blog replicas in small repositories, prioritizing blogs which currently have a high chance of becoming unavailable. A similar line of work is an investigation of DVCMBSs in which there is periodic archiving of blog entries that have been on the service for more than some threshold amount of time. Bandwidth and latency could also be considered as additional criteria when selecting storage hosts. Nodes may also donate CPU and RAM to support additional functionality of the DVCMBS. Another interesting extension regards the energy efficiency of different directory service policies.

Acknowledgments

This work has been partially supported by the Erasmus+ programme (2017-1-ES01-KA103-036672), and the Spanish Ministry of Science, Innovation and Universities (PGC2018-097599-B-I00).

References

[1] Statista, "Number of monthly active twitter users worldwide from 1st quarter 2010 to 1st quarter 2019," https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/, 2019, [Online; accessed 18-May-2020].

[2] ——, "Cumulative total of tumblr blogs from may 2011 to april 2019," https://www.statista.com/statistics/256235/total-cumulative-number-of-tumblr-blogs/, 2020, [Online; accessed 18-May-2020].

[3] S. Oukemeni, H. Rifà-Pous, and J. M. Marquès Puig, "Ipam: Information privacy assessment metric in microblogging online social networks," *IEEE Access*, vol. 7, pp. 114 817–114 836, 2019.

[4] S.-W. Seong, J. Seo, M. Nasielski, D. Sengupta, S. Hangal, S. K. Teh, R. Chu, B. Dodson, and M. S. Lam, "Prpl: a decentralized social networking infrastructure," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010, p. 8.

[5] D. P. Anderson and G. Fedak, "The computational and storage potential of volunteer computing," in *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, vol. 1. IEEE, 2006, pp. 73–80.

[6] V. Darlagiannis, "Hybrid peer-to-peer systems," in *Peer-to-Peer Systems and Applications*. Springer, 2005, pp. 353–366.

[7] A. Grasas, A. A. Juan, J. Faulin, J. de Armas, and H. Ramalhinho, "Biased randomization of heuristics using skewed probability distributions: a survey and some applications," *Computers & Industrial Engineering*, vol. 110, pp. 216–228, 2017.

[8] T. M. Mengistu and D. Che, "Survey and taxonomy of volunteer computing," *ACM J. Comput. Surv*, pp. 1–35, 2019.

[9] J. Panadero, J. de Armas, X. Serra, and J. M. Marquès, "Multi criteria biased randomized method for resource allocation in distributed systems: Application in a volunteer computing system," *Future Generation Computer Systems*, vol. 82, pp. 29 – 40, 2018.

[10] J. Panadero, L. Calvet, J. M. Marquès, and A. A. Juan, "A simheuristic approach for resource allocation in volunteer computing," in *2017 Winter Simulation Conference (WSC)*, Dec 2017, pp. 1479–1490.

[11] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe, and G. Figueira, "A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems," *Operations Research Perspectives*, vol. 2, pp. 62 – 72, 2015.

[12] Y. Alsenani, G. Crosby, and T. Velasco, "Sara: A stochastic model to estimate reliability of edge resources in volunteer cloud," in *2018 IEEE International Conference on Edge Computing (EDGE)*, July 2018, pp. 121–124.

[13] T. M. Mengistu, D. Che, A. Alahmadi, and S. Lu, "Semi-markov process based reliability and availability prediction for volunteer cloud systems," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 359–366.

[14] T. Mengistu, D. Che, and S. Lu, "Multi-objective resource mapping and allocation for volunteer cloud computing," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 2019, pp. 344–348.

[15] Y. Alsenani, G. V. Crosby, T. Velasco, and A. Alahmadi, "Remot reputation and resource-based model to estimate the reliability of the host machines in volunteer cloud environment," in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2018, pp. 63–70.

[16] M. Selimi, L. Cerdà-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, and J. Crowcroft, "A lightweight service placement approach for community network micro-clouds," *Journal of Grid Computing*, vol. 17, no. 1, pp. 169–189, Mar 2019.

[17] H. Guler, B. B. Cambazoglu, and O. Ozkasap, "Task allocation in volunteer computing networks under monetary budget constraints," *Peer-to-Peer Networking and Applications*, vol. 8, no. 6, pp. 938–951, Nov 2015.

[18] T. Ghafarian and B. Javadi, "Cloud-aware data intensive workflow scheduling on volunteer computing systems," *Future Generation Computer Systems*, vol. 51, pp. 87 – 97, 2015.

[19] T. Ghafarian, H. Deldari, B. Javadi, M. H. Yaghmaee, and R. Buyya, "Cycloidgrid: A proximity-aware p2p-based resource discovery architecture in volunteer computing systems," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1583 – 1595, 2013.

[20] K. Ramachandran, H. Lutfiyya, and M. Perry, "Decentralized approach to resource availability prediction using group availability in a p2p desktop grid," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 854–860, 2012.

[21] K. Graffi, D. Stingl, C. Gross, H. Nguyen, A. Kovacevic, and R. Steinmetz, "Towards a p2p cloud: Reliable resource reservations in unreliable p2p systems," in *2010 IEEE 16th International Conference on Parallel and Distributed Systems*. IEEE, 2010, pp. 27–34.

[22] V. Šešum-Čavić, E. Kühn, and L. Fleischhacker, "Efficient search and lookup in unstructured p2p overlay networks inspired by swarm intelligence," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019.

[23] Z. Tienan, "A novel replication model with enhanced data availability in p2p platforms," *International Journal of Grid and Distributed Computing*, vol. 9, no. 4, pp. 151–160, 2016.

[24] X. Meng, "A churn-aware durable data storage scheme in hybrid p2p networks," *The Journal of Supercomputing*, vol. 74, no. 1, pp. 183–204, 2018.

[25] P. Xiao and T. Zhang, "A novel replication scheme based on prediction technology in virtual p2p storage platform," *International Journal of Networking and Virtual Organisations*, vol. 20, no. 1, pp. 90–103, 2019.

[26] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Ö. Özkasap, "Decentralized and locality aware replication method for dht-based p2p storage systems," *Future Generation Computer Systems*, vol. 84, pp. 32–46, 2018.

[27] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Öznur Özkasap, "Decentralized utility- and locality-aware replication for heterogeneous dht-based p2p cloud storage systems," 2019.

[28] S. Caron, F. Giroire, D. Mazauric, J. Monteiro, and S. Pérennes, "P2p storage systems: Study of different placement policies," *Peer-to-Peer Networking and Applications*, vol. 7, no. 4, pp. 427–443, 2014.

[29] K. S. Puthusseri, A. Shetty, M. Patil, and K. K. Devadkar, "Storage and availability aware fragment placement for p2p storage systems," in *2019 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2019, pp. 0867–0871.

[30] E. Spaho, L. Barolli, and F. Xhafa, "Data replication strategies in p2p systems: A survey," in *2014 17th International Conference on Network-Based Information Systems*. IEEE, 2014, pp. 302–309.

[31] B. Mohammadi and N. J. Navimipour, "Data replication mechanisms in the peer-to-peer networks," *International Journal of Communication Systems*, vol. 32, no. 14, p. e3996, 2019.

[32] T. Duong-Ba, T. Nguyen, B. Bose, and D. A. Tran, "Distributed client-server assignment for online social network applications," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 4, pp. 422–435, 2014.

[33] L. Zhang and X. Tang, "The client assignment problem for continuous distributed interactive applications: Analysis, algorithms, and evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 785–795, 2014.

[34] C. Diot and L. Gautier, "A distributed architecture for multiplayer interactive applications on the internet," *IEEE Network*, vol. 13, no. 4, pp. 6–15, 1999.

[35] H. Zheng and X. Tang, "The server provisioning problem for continuous distributed interactive applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 271–285, 2016.

[36] H. Nishida and T. Nguyen, "Optimal client-server assignment for internet distributed systems," in *Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN 2011)*, vol. 1, 2011, pp. 1–6.

[37] P. Sobkowicz, M. Thelwall, K. Buckley, G. Paltoglou, and A. Sobkowicz, "Lognormal distributions of user post lengths in internet discussions - a consequence of the weber-fechner law?" *EPJ Data Science*, vol. 2, no. 2, pp. 1–20, 2013.

[38] B. Javadi, D. Kondo, J. Vincent, and D. P. Anderson, "Mining for statistical models of availability in large-scale distributed systems: An empirical study of seti@home," in *2009 IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, 2009, pp. 1–10.

[39] J. Banks, I. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*. Pearson, 2005.

A Results for the case of gamma distributed online duration and log-normal distributed offline duration

The experiments of Tables 1 and 2, and Figure 5, have been repeated for the case where node online duration follows a gamma distribution and node offline duration follow a log-normal distribution. These distributions were found to be good fits in [38].

The mean offline duration of nodes are uniformly distributed in the range $[2.5, 100]$ per time interval. This range corresponds to arrival rates in the range $[0.01, 0.4]$, the same range as that used for the case where arrivals were modeled using an exponential distribution. The standard deviation parameters are uniformly distributed in the range $[7.07, 100]$, matching the variance of the case where offline duration was modeled according to an exponential distribution. The gamma distributed computer/blogger online sessions have mean duration uniformly distributed in the range $[1, 10]$ per time interval and standard deviations uniformly distributed in the range of $[1, 10]$. These are the same ranges as those used for the case where online duration followed a log-normal distribution. This approach is designed to allow the maximum possible level of comparability between the experiment results reported in the paper and those in this appendix.

Table 4 provides details of the non-dominated solutions for the case where node online duration follow a Gamma distribution and node offline duration follow a Log-normal distribution.

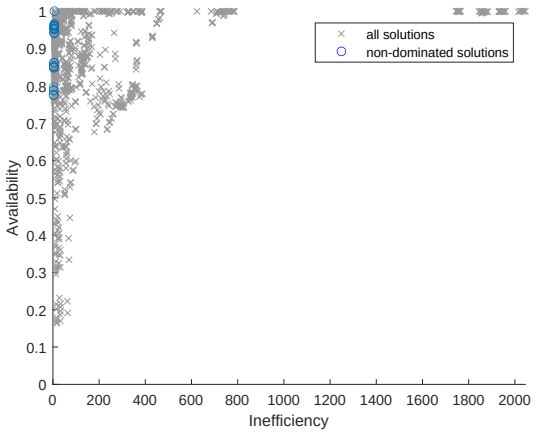

Figure 11: Availability and inefficiency measures for each combination of directory service configuration and replication policy parameters.

	Input decision	Parameter description	Treatment level (availability (inefficiency))		
			Low	Medium	High
directory service configuration options	R1 $\{false, true\}$	Replicate leaver's repository entries	0.827 (81.684)	-	0.943 (263.997)
	R2 $\{false, true\}$	Store replicas in arrivals repository	0.870 (171.886)	-	0.899 (173.796)
	V1 $\{false, true\}$	Validate replicas using message counts	0.844 (262.167)	-	0.926 (83.515)
	V2 $\{false, true\}$	Validate replicas using online replicas	0.836 (298.256)	-	0.933 (47.426)
	V3 $\{false, true\}$	Update out-of-date replicas	0.882 (172.863)	-	0.887 (172.818)
	V4 $\{false, true\}$	Retain replicas until validity proven	0.882 (172.963)	-	0.888 (172.719)
Replication policy parameters	N $\{1, 2, 4\}$	Target replicas per blog	0.800 (132.797)	0.916 (184.971)	0.939 (200.754)
	M $\{1, 2, 4\}$	Max replicas stored in arrivals repository	0.880 (172.039)	0.885 (172.687)	0.889 (173.796)
	P $\{0, 0.5, 1\}$	Target probability that one replica remains	0.787 (35.984)	0.920 (182.3)	0.948 (300.238)
	F $\{0, 0.5, 1\}$	Return time estimation quantile parameter	0.874 (78.293)	0.893 (194.855)	0.887 (245.374)
	β $\{0, 0.5, 1\}$	Biased randomization parameter	0.893 (153.668)	0.882 (180.585)	0.879 (184.269)

Table 3: Directory service options impact on average replica availability and inefficiency.

Non-dominated solutions													
Input parameters											Performance measures		
R1	R2	V1	V2	V3	V4	N	M	P	F	β	Availability	Inefficiency	Average available replicas per blog
1	0	1	1	0	1	2	1	0	0.500	1	0.999	6.290	4.732
0	1	1	1	0	1	2	4	0	0.500	0	0.965	6.127	3.955
0	1	1	1	0	1	2	4	0	0	0	0.963	6.108	3.939
0	1	1	1	0	0	2	2	0	0	0	0.961	6.020	3.831
0	1	1	1	0	0	2	4	0	0.500	0.500	0.959	5.885	4.082
0	1	1	1	1	1	2	4	0	0.500	1	0.958	5.817	4.091
0	1	1	1	0	1	2	4	0	0.500	1	0.957	5.770	4.045
0	1	1	1	0	0	2	2	0	0.500	0.500	0.953	5.725	3.919
0	1	1	1	0	1	2	2	0	0.500	0.500	0.952	5.689	3.928
0	1	1	1	1	1	2	2	0	0.500	1	0.950	5.654	3.933
0	1	1	1	0	1	2	2	0	0.500	1	0.950	5.599	3.876
0	1	1	1	0	1	2	1	0	0.500	0.500	0.941	5.414	3.688
0	1	1	1	1	1	2	1	0	0.500	1	0.941	5.374	3.692
0	1	1	1	0	1	2	1	0	0.500	1	0.940	5.305	3.627
0	0	1	1	1	1	2	1	0.500	0	0	0.861	4.830	2.942
0	0	1	1	0	1	2	1	0.500	0	0	0.861	4.765	2.903
0	0	1	1	1	1	2	1	0	1	0	0.853	4.144	2.489
0	0	1	1	1	1	2	1	0	0.500	1	0.851	4.001	2.665
0	0	1	1	1	1	2	1	0	0.500	0.500	0.848	3.992	2.664
0	0	1	1	0	1	2	1	0	0.500	1	0.847	3.837	2.521
0	0	1	1	1	1	1	1	0	0.500	1	0.795	3.209	1.671
0	0	1	1	1	1	1	1	0	1	0	0.787	3.199	1.575
0	0	1	1	1	1	1	1	0	0	0	0.786	3.168	1.567
0	0	1	1	0	1	1	1	0	1	0	0.775	3.078	1.351
0	0	1	1	0	1	1	1	0	0	0	0.774	3.075	1.352

Table 4: Directory service options impact on availability, inefficiency and average available replicas per blog.